(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 193 940 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.04.2002 Bulletin 2002/14

(51) Int Cl.7: H04L 29/06

(21) Application number: 01118847.1

(22) Date of filing: 21.03.1995

(84) Designated Contracting States:
DE GB NL

(30) Priority: 21.03.1994 US 215849

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
95301865.2 / 0 674 414

(71) Applicant: AVID TECHNOLOGY, INC.
Tewksbury, MA 01876 (US)

(72) Inventors:
- Hamilton, James
  Carlisle, Massachusetts 01741 (US)
- Hennessy, Richard
  Nashua, New Hampshire 03063 (US)
- Choa, Pong
  Cambridge, Massachusetts 02139 (US)
- Chopde, Avinash
  Andover, Massachusetts 01810 (US)
- Peters, Eric
  Carlisle, Massachusetts 01741 (US)
- Koutavas, Jay
  Nashua, New Hampshire 03060 (US)

(74) Representative: Kazi, Ilya et al
Mathys & Squire,
100 Gray's Inn Road
London WC1X 8AL (GB)

Remarks:
This application was filed on 14 - 08 - 2001 as a divisional application to the application mentioned under INID code 62.

(54) Apparatus and computer-implemented process for providing real-time multimedia data transport in a distributed computing system

(57) The problems of latency, flow control and data loss, and data movement within a client system memory are solved in a distributed multimedia system so as to enable real-time transmission of broadcast quality media data over the network. Latency is reduced by an estimation by the server of client needs. Data loss is prevented and flow control is provided by permitting the server to send only as much information as the network interface can reliably receive. Data movement is minimized by copying data directly from the network interface to memory in a display processor or audio processor.

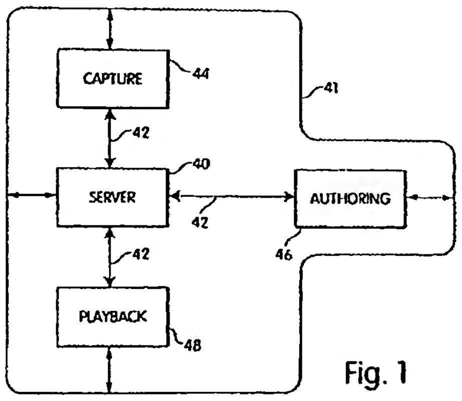


Fig. 1

Description

FIELD OF THE INVENTION

[0001] This invention is related to systems and processes for maximizing media data transfer rates in distributed multimedia computer systems.

[0002] More particularly, this invention relates to systems and processes for preventing loss of media data during media data transfer between a client and a server in a distributed multimedia system.

[0003] This invention also more particularly relates to systems and processes for retrieving media data with minimum latency from a disk-based media data server such that a client multimedia application has access to sufficient broadcast quality media data.

BACKGROUND OF THE INVENTION

[0004] A variety of kinds of digital multimedia computer systems are now available. Multimedia systems typically use digital audio and digital image data, especially digital video. It should be understood that in this context, digital video means a sequence of two-dimensional digital still images without any synchronization or other timing information typically associated with a video signal such as a television signal. These systems generally fall into one or more of three broad categories: capture, authoring, and playback.

[0005] Capture involves generating digital media from typically analog media. For example, video signals on a video tape or on film are converted into a sequence of two-dimensional digital still images. These images are typically compressed using known compression systems so as to reduce storage requirements.

[0006] Authoring systems, such as the AVID/1 Media Composer, commercially available from Avid Technology, Inc. of Tewksbury, Massachusetts, allow a user to generate multimedia programs, for example by editing source material or by generating information like computerized graphics. Portions of the source material are called clips. For example, multiple takes of a scene in a film could be captured. An authoring system could be used to edit clips from these takes into an appropriate final version of the scene.

[0007] Playback systems are used to display a multimedia program to a viewer. The program typically is previously edited by an authoring system using previously captured material. The display medium may be, for example, broadcast television or conversion to video signals to create a video tape or video disk, a computer display, etc. Thus, playback generally involves retrieval and transmission of the digital media from storage to the display location according to the specified program and may involve conversion of the digital media to an analog form. Very often an authoring system includes a form of playback system.

[0008] In some industries, all three of these kinds of systems are used in day-to-day operations. For example, in a broadcast or cable television system, media is captured from satellite, cable or another source. A program is then edited from this material. For example, video and audio footage of news events from around the world may be transmitted via satellite and captured by a broadcast station, which then prepares a news program. The final news program is then played back for broadcast. As another example, in the film industry, media may be captured, and then played back by a director to determine whether any more takes of a given scene would be required. Authoring systems are also becoming more commonly used to facilitate the editing process of a film. After a digital version of the film is completed, this version is used to create a final film product.

[0009] In facilities where there may be many users of the same multimedia material, using multiple types of multimedia systems, a need has arisen to provide these capabilities in a distributed computing environment which allows real-time access to and viewing of multimedia information. Real-time access signifies data rates which support providing broadcast quality video at a client. That is, real-time access involves the ability to play, cut and paste, view a single frame and provide fast forward (or reverse) access to video information at a server. This also may signify an ability to recover from a worst-case disk access. Generally, data transfer rates of about five (5) megabytes per second (Mbps) are preferable. Real-time audio and video systems also require large amounts of data to be transferred. These transfer rates and amounts should be achieved without loss and with minimal latency.

[0010] To this day, no systems are commercially-available to handle adequately the data transfer requirements of such multimedia systems, particularly broadcast quality video. Because such distributed systems are not currently commercially-available, users and television stations or film studios or post-production facilities typically have to transport disks containing the multimedia information among different machines in order to perform the various capture, authoring, and playback activities, which is generally undesirable. Alternatively, all of these activities are performed on a single machine.

[0011] For other kinds of applications, there are several conventional methods for transferring large amounts of data between a server and a client in a distributed computer system over a network interconnection. In a first method, a server transmits the data, in bulk, directly to a client. Traditional distributed computing systems utilize a widespread industry practice of a disk server system coupled with an in-memory data cache to provide a mechanism that a client may use to read data. The file system of the underlying operating system performs larger than requested disk reads in an attempt to fill the cache with information that the system estimates the client will need next. One problem with this method is that the client has no control over selection of data, or timing of the transfer.

[0012] In a second method, the client periodically and unpredictably makes requests for data from the server. One problem with this method is that a delay (latency) is incurred while the server prepares the data for delivery, for example, by reading it from a local disk into system memory. This class of systems will not provide the minimum latency required by a media data pipeline operating in a variable consumption rate environment within a video and audio processing system.

[0013] Another problem with current systems is that they have not accounted for resource limitations along the path from the sender to the receiver. That is, when data, such as audio or video media data, is transmitted over a network, a flow control mechanism is required so that the sender does not transmit the data faster than tha receiver can accept it. Otherwise, portions of the data packets may ba lost in transmission even though the physical interconnect has not failed in any way. This packet loss is normally handled by retransmission, but the real-time requirements of multimedia playback, especially in a broadcast environment, preclude retransmission as a solution.

[0014] The Internet protocol, Transmission Control Protocol/Internet Protocol (TCP/IP), is an example of current networking industry practice. TCP/IP is an example of the use of an end-to-end window extension technique. The receiver extends a window to the sender. The sender sends until this window is consumed. The window only refers to the amount of protocol data unit (PDU) buffer memory available to the application. Network interface level buffering is not considered. If the buffers in the network interface are overflowed or congested, PDUs will be lost. TCP/IP was designed to operate in the face of a PDU loss on the interconnect between the sender and the receiver. Tha usa of timeouts and sequence number checking on each message allows the sender and receiver to determine that a PDU or acknowledge message has been lost or corrupted while moving across the network from such a loss. Transmission retry is used to recover. The ovarhead of the retry and timeouts reduces the possible data transfer rate below that needed to support processing broadcast quality video and audio information.

[0015] Another common problem in distributed systems is that the receiver typically experiences significant overhead in identifying a final destination in user-level memory into which data is to be copied from an intermediate system buffer and the network interface. This overhead reduces the possible data transfer rate to levals that make impossible the movement of broadcast quality video and audio media data at sufficiant speed.

SUMMARY OF THE INVENTION

[0016] The present invention ovarcomes the problems of latency, flow control and data loss, and minimizes data movement within a client system memory in a distributed multimedia system so as to enable real-time transmission of broadcast quality media data over the network.

[0017] Latency is reduced by an estimation by the server of client needs. Data loss is prevented and flow control is provided by permitting the server to send only as much information as the network interface can reliably receive. Data movement is minimized by copying data directly from tha network interface to memory of input/output davices such as display and audio processors as well as main memory.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] In the drawings,

FIG. 1 is a block diagram of a distributed multi media computing system in accordance with this invention;

FIG. 2 is a more detailed block diagram of the client and server computers of FIG. 1;

FIG. 3 is a block diagram describing a media pipeline as is used in the prior art;

FIG. 4 is a flow chart describing how a server computer prepares to send data to client computer;

FIG. 5 illustrates a window buffer manager at tha server used to anticipate client requests for media data.

FIG. 6 is an illustration of a packet as may be used in accordance with a protocol in this invention;

FIG. 7 is a block diagram of a network interface for a client computer; and

FIG. 8 is a flow chart describing how a client computer receives information.

DETAILED DESCRIPTION

[0019] The present invention will be more completely understood through the following detailed description which should be read in conjunction with the attached drawing in which similar reference numbers indicate similar structures. All references cited herein are hereby expressly incorporated by reference.

[0020] FIG. 1 illustrates a general structure of a distributed multimedia computer system suitable for practicing the invention. It includes a server 40 to which various client systems are interconnected via a high speed media transport 42. The media transport 42 is in a "star" configuration to provide full bandwidth for each client and to support failure recovery. The clients may generally include a capture system 44 and authoring system 46 and a playback system 48 as shown. Many other clients may exist and not all three types of multimedia systems are required.

[0021] To provide low bit error rates, the media transport 42 is preferably optical, e.g. a fiber distributed data interconnect (FDDI) or an asynchronous transfer mode (ATM) system. It should be used only for communicating messages involved in the transfar (i.e., reading and writing) of media data between the client and the server. There should be no other network traffic on the media transport 42.

[0022] To play two-field broadcast video, each client should have about five (5) megabytes/second (Mbps) of data bandwidth for one video stream and four audio streams. This bandwidth allows for three (3) Mbps of two-field, 100Kb/frame compressed video data, 96Kb/sec for each 48 Khz audio channel and recovery bandwidth. Recovery bandwidth is defined as additional bandwidth needed to recover after a disk head move caused by any of a new video file extent access, a need for audio samples for one or more tracks, or spindle contention from two clients accessing the same disk simultaneously. Thus, with current technology an additional bandwidth of 1.5 Mbps is recommended so that the media transport 42 should allow for five (5) Mbps. Highly scalable bandwidth and single point management and. monitoring are also benefits of this kind of interconnect.

[0023] The clients 44, 46, 48 and the server 40 may also be interconnected by a control network 41 which mey be, for example, an Ethernet type of network. The control network 41 is used for management of media files and other resources through services of another system unrelated to the communications of media data. The control network 41 is also suitable for general administrative support.

[0024] As with any network, a connection between the server and the client has to be established before sending any data. The connection initialization process is independent of the protocol for media transport. The connection handshaking is typically done over the conlrof network 41. It may also be done using another protocol on the media transport 42.

[0025] As shown in FIG. 2, the server 40 includes a disk 52 on which the media data is primarily stored. The server 40 also includes a processor 54 and system memories 56 which are interconnected with the disk 52 via a bus 58. This bus 58 is also connected to a network interface 60 which connects to the media transport 42. A client 50 includes a central processor 62 and main memory 64 interconnecled by e bus 66. Also connected to the bus are preferably video memory 63 and audio memory 65, which stores the multimedia data to be used respectively by display processor 67 and audio processor 69 to display video and play audio. This bus connects to a network interface 68 which connects the client 50 to the media transport 42, and thus to the server 40.

[0026] In the embodiment developed by the inventors, the client computers 44, 46, 48 are Macintosh Quadra 950 computers with twenty-eight (28) megabytes (Mb) of memory and Macintosh System 7.1 system software. The server computer is a Silicon Graphics Challenge L computer with 4 cpu, 100 megahertz (Mhz), 256 Mb of memory, 8 gigabytes (gb) of disk storage and the IRIX 5.2 operating system. The media transport 42 and network interface 60 is provided by an optical fiber with the FORE Systems ForeRunner VMA-200 ATM VMEbus Adapter for the server 40, and associated system software, and application program interface (API) library software. For the clients, a FORE Systems ForeRunner NBA-200 ATM NUBUS Adapter was used as the network interface 68, along with associated system software, and API library software. It should be understood that the invention is not limited to these specific machines and components mentioned and that other network Interfaces and computers may be used.

[0027] The need for reading media data from the server 40 into a client is dependent on the need for the media data further down the processing stream, particularly by the display processor and audio processor. That is, this media data should be considered as being used in a sort of media pipeline, such as that described in U.S. Patent No. 5,045,940, issued September 3, 1991 and assigned to Avid Technology, Inc of Tewksbury, Massachusetts. In this system, as shown simplified in FIG. 3, video date is input through an input/output system 100. An audio stream is not shown. The data then goes to a frame buffer 102 from where it is then compressed using a decompression/compression system 104. Compressed data is stored in a buffer 106 from where it is then transferred into permanent storage 108. Conversely, data can be read from permanent storage 108 to the compressed data buffer 106 from where it is decompressed in 104 and applied to a frame buffer 102 from where it is then displayed by an output unit 100. In this system, processing by any one of the subsystems is delayed if the subsystem to which its output is provided cannot receive any more data. The general concept is to maintain the pipeline at a completely full state in all of its stages so as to maximize the quality of the video being seen. The distributed system in this invention would be primarily used so as to provide permanent storage 108 at a server 40 (FIG. 1) and to have the remainder of the subsystems 100-106 at a client. Thus, requests for media data from the server 40 come at times indicated by other subsystems of a media pipeline as media data is needed.

[0028] At the application level, there are two types of messages in this system: protocol requests/responses and media data packets. Protocol requests/responses occur at the application level of the interconnect between the client and the server. Media date transfer occurs at the transport level. Both types of messages are communiceted over the media transport 42.

[0029] Having described a general structure of a distributed computer system and a network protocol for supporting this invention, its use in providing real-time broadcast quality video will now be described. Three general concepts are used to obtain the desired results. First of all, clients send some information to the server 40 in advance of requests for media date, which information allows the server to access the data from the disk and place it in faster system memory in advance of client requests, thus reducing latency. The server, in effect, is allowed to predict the requests from the client. This information is sent by the client to the server as a protocol request/response at the application level. Lossless flow control is provided by having a cliant authorize tha server 40 to send only an amount of data which the natwork interface is capable of handling. Finally, protocol control and identity mechanisms are provided within data packets to be transferred. These protocol control and identity mechanisms are then used by the network interface to process the data and automatically and directly transmit it to an appropriate user-level memory buffer to avoid copying of data into system lavel memory at tha client. These concepts will now be described in more detail.

[0030] How the server is informed in advance of the needs of the client will first be described. In either an authoring system or a playback system, it is simple to predict the media data which will be required by a client. For example, in a typical authoring system, the user selects from a directory a set of clips which will be used in the product to be edited. Upon the selection of these clips, a message may be sent to the server by the client requesting that those clips be prepared for delivery. Similarly, in a playback system, one typically knows in advance all of the clips which will be used in the program to be played. Both of these types of prediction can be readily performed using the Avid/1 Media Composer. For example, when editing using the Avid/1 Media Composer, a user selects a number of clips by selection from a directory. Also, as a program is being edited, a data structure is developed callad a composition which indicates the relationships and ordering of clips used in the program. A list of clips needed can be readily determined and sent to the server to permit the server to prepare in advance the data needed to display these clips.

[0031] The sequence of steps performed to effect this prediction will now be discussad in connection with FIG. 4. First, the list of media objects to be used by the client is created by the client in step 120. The client then requests that the server create a data structure, herein called a TrackList, for storing the media objects for the list in step 122. The TrackList is a resource on the server. Subsequent requests by the client are used to populate, or define tha contents of, the TrackList. The order of these requests dictates the order of the entries in the TrackList, as will be described below. The server responds to the client with an identifier for the TrackList which the client then includes in any requests to the server for associating media objects with the TrackList. The server may refuse a request to create a TrackList due to lack of resources and reply with an error in step 124.

[0032] After an identifier of a TrackList is received, the client then sends, in step 126 a series of requasts to open media files for media objects corresponding to the list created by the client. Each request includes an identifier for the media object to be opened, the identifier of the TrackList, and an indication of whether the media object is to be inserted at the beginning or end of the TrackList. The identifier of the media object includes the media file and a range of bytes within that media file.

[0033] For each request to obtain a media object, the server opens the media object in step 128 and associates it with the TrackList. The operation performed by the server opens a logical read stream to an existing media file, hereinafter called a flow, located on the server. The flow is a cache buffer which is allocated and linked to the TrackList. Thus, the TrackList is a list of flows. It is possible to obtain single frame access to media files by specifying a NULL identifier for the TrackList. In this instance, no resourca reservations are mada.

[0034] If this server contains the media fila identifiad by the client request, it returns an identifier to the flow that was opened by it in step 130, which identifier the client includes with subsequent requests to read media data from this file. If a requested media fila is not located on the server an error is returned. If an incorrect identifier for tha TrackList is provided, an arror is also returned. If the media file requested is in use by another client, an error is also issued due to a resource conflict.

[0035] After tha appropriate requests to open media files for read access, the media data desired by a client for a track are placed in a window buffer (described below) associated with that track so that the media data are ready to be accessed upon a read request. Using the predictive information in the TrackList the server will already have read the needed media data into the window buffer and can immediately fulfill the request.

[0036] As shown in FIG. 5, a window manager 140 at the server 40 is responsible for maintaining a window about a currently active pointer 144 in the window buffer 142 by doing autonomous read forward or backward operations. Reading sample data with minimum server turn-around delay is thus obtained because the server can do look ahead disk reads to fill the window buffer 142 for the next flow while the current flow is still being read by client. The TrackList allows the server to autonomously look ahead according to the information about the order of play of the streams, thus maintaining this window buffer 142 full. These read ahead operations for the media objects referenced by the TrackList can be performed such that the window buffer 142 will always contain the exact data needed to satisfy subsequent read operations.

[0037] When the media data is prepared in the window buffer by the server, the server 40 is ready to handle read requests from the client, which will now be described in connection with Figs. 6-10. The preparation activities are performed at the application level; mechanisms for reading are performed at the transport level as will be described below.

[0038] To enhance throughput for reading media data in this system, a new network interface and protocol was developed for use at the transport level.

[0039] As shown in FIG. 7, the network interface 68 includes a memory 80, preferably a first-in, first-out (FIFO) memory for receiving packets from the media transport 42. The memory 80 is controlled by a controller 82 which reads packets from the memory 80, and then transfers the data portion of those packets, thus demultiplexing the packets, to logically separate direct memory access (DMA) elements 84, 86 and 88 which are connected to the system bus 66. Media data of a video track goes to the logical DMA to video memory 66. Similarly, audio data is sent to the logical DMA to audio memory 88. Finally, data concerning protocol requests/ responses and window packets are sent to the logical DMA to main memory 84.

[0040] As discussed above, the client creates a TrackList for each track in the media program to be played, authored or captured. Likewise, a receive channel is also created by the client for each of these tracks to receive media data. The server similarly opens the same number of transmit channels, each associated with a TrackList. A channel is a logical construct used to control flow of data between the network interface and main memory or audio or video memory. More particularly, a channel is a half-duplex virtual circuit with a logical DMA control scheme, a flow control scheme to be described below and associated state information as discussed below. Each channel has a separate logical DMA controller. Thus the interface of FIG. 7 represents two media data channels. Both the server and the client have a similar command channel which is used for communicating data concerning protocol requests and responses. Each transmit channel at the server is organized as a cache, or a window buffer, into the stream of bytes being read by the client as discussed above in connection with FIG. 5.

[0041] The network interface maintains a set of state information for each active receive channel. This information includes 1) an offset in bytes from the start of a packet to the start of the field storing the identifier for the receive channel, 2) an indication of the length in bytes of the identifier of this receive channel, 3) the identifier used by the network interface to identify this receive channel, 4) an offset in bytes from the start of a packet to the start of data, 5) a starting address of the destination memory to which the network interface will move the data, 6) a queue of all receive requests for this channel and including the address and size of all transfer units to be sent authorization indicating the number and size of all transfer units authorized to be sent to this receive channel (to be described below), and 7) a count in bytes of data transferred by the network interface for this receive channel to this point in time. For each channel items 1), 2) and 4) are long lasting and should not change in real time. The network interface maintains state variables for the remainder. The network interface also maintains a global state variable indicative of an amount of data that it has authorized to be received.

[0042] The network interface also maintains a set of state information for each active transmit channel. This information includes 1) an offset in bytes from the start of a packet to the start of the identifier for a channel, 2) an indication of the length in bytes of the identifier, 3) an indication of the token used by the network interface to identify the transmit channel, 4) an offset in bytes from the start of the packet to the start of data, 5) a starting address of the source buffer from which the network interface will move the data, 6) an offset in the source buffer from where the media data will be taken, 7) an indication of the total number of data bytes to transfer from the indicated address, and 8) stream protocol header information, used by the network interface to encapsulate the retrieved data bytes into a proper network packet. The network interface also keeps track of the number and size of transfer units authorized for each transmit channel and the number of outstanding bytes authorized to be transmitted to a given destination address.

[0043] This new network interface was defined by using a protocol handler which is part of a device driver for the network interface. The protocol handler demultiplexes received packets on the basis of protocol type. An indication of the protocol type of a packet, often called a discriminator, is located in a predetermined location in the packet. A typical packet for this new protocol is shown in FIG. 4.

[0044] In this protocol, as shown in FIG. 4, media data is encapsulated within a packet 70. This format includes a network header 72, a protocol header 74, data 76, and a packet or package check sum 78. The network header 72 includes the indications of the sender and receiver of the packet, as well as the discriminator indicating the protocol type. The protocol header indicates the type of packet within the protocol being received. At the transport level, there are two types of packets for media transport in this protocol: data packets and window packets.

[0045] Data packets contain an arbitrary amount of data. Such packets may be media data packets for media channels or protocol request/responses data for command channels. A data packet contains in the protocol header 74 an indication of the receive channel at the destination and, in data portion 76, a block of data destined for the receive channel indicated. The amount of data in the packet and the offset in bytes in the destination buffer where the network interface will move the data are also included in the protocol header 74.

[0046] Window packets indicate information used by the server to control the amount of data sent to the client. A window packet includes an indication of the channel at the source of the window packet to which data is to be sent (i.e., the destination of the requested data) an indication of whether an additional transfer unit is authorized, as well as a number of bytes which is authorized to be sent. A window packet authorizes the server to transmit either one additional transfer unit on a given virtual channel or N additional bytes on the network interface, or both. The first of these allows the source of the window packet to prevent the transmission of packets for which no receive buffers are available. The second ensures that the flow of data does not overrun the capabilities of the network interface card to buffer packets for delivery to the source of the window packet. How these window packets are used will be described in more detail below.

[0047] The network interface (FIG. 7) is used in the following manner to control flow between the sender or server 40 and client 50. First, as mentioned above for each track, a virtual receive channel is created. That is, for example, a video program typically has at least one track of video data and one track of audio data. A virtual receive channel is created for each of these tracks by the client in step 90 of FIG. 8. For each virtual receive channel at the client a logical DMA is created in the network interface 68. When the client places a request to the server for media data, an identifier of the receive channel to which media data will be sent is included, along with an indication of the media file to be accessed and an amount of data requested from it, called a transfer unit, is also provided. This request is made at the application level after the TrackList for the channel is established. Next, at the transport level, a window packet is sent in step 92, authorizing one transfer unit to be sent to this receive channel. However, the virtual receive channel only has a given size S, i.e., a maximum amount of data that it may hold. Thus, the window packet authorizing a transfer unit only authorizes a number of bytes to be sent which is less than the size S of the virtual channel. Typically, the first transfer for a program includes both, as identified above. However, if an application is playing a number of media files, a media file in the middle of the program may be requested while another is already in memory at the client. Thus, a transfer unit may be authorized by such a request, but no data in addition to that authorized previously may be sent. This method assumes that there is no other traffic on the network interface that is under its control. It also assumes the existence of a network interface 68 that is capable of reliably buffering at least one complete data packet at the full speed of the network.

[0048] The network interface processes a transmit channel by reading network packet size data chunks from the specified addresses, with the last chunk being of arbitrary length, while encapsulating the data in proper header information then sending completed packets to the destination node. The network interface only reads as many bytes as specified. The data packet sent also includes the identifier of the receive channel previously provided by the client.

[0049] The received data is then processed by the client in step 94.

[0050] For reception, the data must be separated from the packet and moved to buffers for use. Traditionally this separation is performed by the client after the packet has been moved into system memory by the network interface. The client then moves the data to the appropriate destination using a software copy loop. However, no system memory buffer copy is possible given the data rates required of broadcast quality video. This traditional method is bypassed in this invention to enhance throughput by minimizing data copying using direct memory access. The identifier of the receive channel is used by the network interface under direct memory access (DMA) control to directly transfer the media data from the network interface into the system memory or other i/o device memory, such as video memory 63 or audio memory 65, thus enhancing throughput, as shown in FIG. 7. After the received data is processed, a window authorization is sent again for more data, in step 92. Thus, a series of network packets is received into the network interface are decomposed from the packets then written directly from the network interface to specific destination memory addresses with media data processing hardware. The network interface uses the state variables of the receive data channel to move the data portion of the packet into the destination addresses.

[0051] As an example, assume that a client wishes to have one megabyte of information from a media file. A read request is sent from the client to the server at the application level to request that amount of data. The one megabyte of information represents a transfer unit. Next, a window authorization is then sent to the server at the transport level authoring one transfer unit to be sent, but that only 16 kilobytes of data to be sent, assuming that the size of the packet FIFO (80 in FIG. 7) at the client is 16 kilobytes. The server then sends the appropriate data from these senders transmit stream only sending a series of packets up to 16 kilobytes. One packet is likely not to be a total of 16 kilobytes, it may be only a small number, e.g., typically approximately 4K bytes. As the client processes the received data, at some point, another window authorization may be sent, for example, authorizing an additional 8K of bytes to be sent. The client will do this after a certain number of bytes of buffer memory in the network interface become available. The server continues to send media data until the number of bytes authorized is exhausted or until the requested transfer unit is exhausted.

[0052] Aside from providing real-time access to broadcast quality media data, this system provides numerous advantages over standard multimedia systems. For example, recording digitized media directly onto the server allows a user at a client computer to play video without having to copy large media files from the server to the client computer. Also, local disk space at each client computer is not needed to hold copies of media files. One copy of each file at the server is all that is needed.

[0053] Having now described a few embodiments of the invention, it should be apparent to those skilled in the art that the foregoing is merely illustrative and not limiting, having been presented by way of example only. Numerous modifications and other embodiments are within the scope of one of ordinary skill in the art. For example, fault tolerance may be provided in this system by providing redundancy, for example, redundant interconnect and network subsystems can be added as desired to improve system robustness. These and other modifications are contemplated as falling within the scope of the invention as defined by the appended claims.

[0054] The invention may also be defined by the following numbered clauses.

1. A distributed multimedia network system, comprising:

   a server computer having a disk on which media data is stored;
   a client computer having means for requesting media data from the server computer and means for providing to the server computer a list of media data which may be requested by the client;
   a media transport connecting the server computer to the client computer and communicating media data between the client computer and the server computer; and
   in the server computer, means for retrieving media data from the disk as indicated in the list provided by the client computer and for caching the retrieved media data.

2. The distributed multimedia network system of clause 1,

   wherein the server computer includes means for sending to the client computer an identifier which references the cached media data at the server,
   wherein the client computer further comprises means for sending requests to the server, with the identifier, for a next portion of the media data, and
   wherein the server computer, in response to said requests, sends the requested media data to the client computer.

3. The distributed multimedia network system of clause 2,

   wherein the means for retrieving, in the server computer, only retrieves a first portion of the media data in the list at a given time, and
   wherein the means for retrieving, in the server computer, retrieves a latter portion of the media data in the list after the client computer requests the first portion of the media data.

4. The distributed multimedia network system of clause 2,

   wherein the requests from the client include an indication of an amount of data to be transferred, and
   wherein the client computer includes means for determining the amount of data to be transferred according to capacity of a network interface in the client computer.

5. The distributed multimedia network system of clause 1,

   wherein the client computer includes a network interface,
   wherein the server computer includes means for sending cached media data to the client computer including an indication of an application at the client computer to receive the media data, and
   wherein the network interface of the client computer includes means for receiving the media data from the server computer and for transmitting the media data directly to memory in the client computer used by the application without copying the media data into system level memory in the client computer.

6. The distributed multimedia network system of clause 1,

   wherein the server computer maintains a window buffer in which media data is cached, and
   wherein the means for retrieving and for caching retrieves and caches only enough media data to fill the window buffer.

7. The distributed multimedia network system of clause 6,

   wherein the server computer, in response to requests from the client, includes means for sending media data in the window buffer to the client computer, and
   wherein the server computer includes means for operating the means for retrieving and caching so as to fill the window buffer after sending media data from the window buffer to the client computer.

8. The distributed multimedia network system of clause 1, wherein the media data is full-motion broadcast quality video.

9. The distributed multimedia network system of clause 8, wherein the media data also includes audio associated with the video.

Claims

1. A process for directing video and audio data from a computer network to output devices, comprising:

   associating a video output device having a

memory with a first channel having a direct memory access engine;
associating an audio output device having a memory with a second channel having a direct memory access engine;
receiving data packets from a computer network into a buffer, wherein each data packet includes header information and data, including an indication of one of the first and second channels; and
for each data packet, transferring the data in the data packet directly from the buffer to a memory associated with the output device corresponding to the indication of the first and second channels using the direct memory access engine of the channel, thereby bypassing copying of the data by the operating system to system level memory.

2. A network interface for directing video and audio data from a computer network to output devices, comprising:

means for associating a video output device having a memory with a first channel having a direct memory access engine;
means for associating an audio output device having a memory with a second channel having a direct memory access engine;
means for receiving data packets from a computer network into a buffer, wherein each data packet includes header information and data, including an indication of one of the first and second channels; and
means for transferring the data in each data packet directly from the buffer to the memory of the output device corresponding to the indication of the first and second channels using the direct memory access engine of the channel, thereby bypassing copying of the data by the operating system to system level mamory.

3. A network interface for directing video and audio data from a computar natwork to output devices, comprising:

a first channel having a direct memory access engine for accessing memory in a video output device;
a second channel having a diract mamory access engine for accessing memory in an audio output device;
a buffer for receiving data packets from a computer network, wherein each data packet includes header information and data, including an indication of one of the first and second channels; and
a controller for transferring the data in each data packet directly from the buffer to the memory of the output device corresponding to the indication of the first and second channels using the direct memory access engine of the channel, thereby bypassing copying of the data by an operating system to a system level memory.

4. A process for controlling flow of data over a computer network from a server to a client, comprising:

sending a request to a server indicating a channel at the client, a file to be accessed and an amount of data to be read from the file;
sending a message to a server authorizing data to be transfarred from the file in an amount lass than an amount of space available in a buffer on the network interface;
receiving data packets from the server and storing the data packets into the buffer, wherein each data packet includes an indication of the channel at the client;
processing the received data to provide the received data from the buffer to a device associated with the indicated channel; and
repeating staps of sending the message, receiving data packets and processing tho received data until the amount of data to be read from the file is received.

5. The process of claim 4, wherein the data comprises video and audio data to be played back on respective output devices, wherein the process further comprises:

associating a video output device having a memory with a first channel having a direct memory access engine;
associating an audio output device having a memory with a second channel having a direct memory access engine;

wherein the indication of the channel includes an indication of one of the first and second chanals; and
wherein the step of processing the each received data packet comprises transferring the data in the data packet directly from the buffer to a memory associated with the output device corresponding to the indication of the first and second channels using tha direct memory access angine of the channel, thareby bypassing copying of the data by the operating system to system level memory.

6. The process of claim 5, further comprising:

sending the server a list of files which may be accessed by the client;
receiving from the server an indicator of a re-

source allocated at tha server for storing the data of each of the files; and

wherein tha raquest sent to the server to read an amount of data from a file includes the indicator of the resource allocated for storing the data of the file.

7. The process of claim 4, further comprising:

sending the server a list of files which may be accessed by the client;
receiving from the server an indicator of a resource allocated at the server for storing the data of aach of the files; and

wherein tha request sant to the servar to read an amount of data from a fila includes the indicator of the resource allocated for storing the data of the file.

8. An apparatus for controlling flow of data over a computer network from a server to a client, comprising:

means for sending a request to a server indicating a channel at the client, a file to be accessed and a dasirad amount of data to be read from the file;
means for sending a message to a server authorizing data to be transferred from the file in an amount less than an amount of space available in a buffer on the network interface;
means for receiving data packets from the server and storing the data packets into tha buffer, wherein each data packet Includes an indication of the channel at the client;
means for processing the received data to provide the received data from the buffer to a device associated with the indicated channel; and

wherein the means for sending the message, means for receiving data packets and maans for processing the recelved data cooperate to causa the desired amount of data to be received from the file.

9. The apparatus of claim 8, wherein the data comprises video and audio data to be played back on respective output devices, wherein the apparatus further comprises:

means for associating a video output device having a memory with a first channel having a direct memory access engine;
means for associating an audio output device having a memory wlth a second channel having a direct memory access engine;

wherein the indication of the channel includes an Indication of one of tha first and second channels; and
wherein the means for processing the each received data packet comprises means for transferring the data in the data packet directly from tha buffer to a memory associated with the output device corresponding to tha indication of the first and sacond channals using tha diract memory access engine of the channel, thereby bypassing copying of the data by the operating system to system level memory.

10. The apparatus of claim 9, further comprising:

means for sending the server a list of files which may be accassed by tha client;
means for receiving from the server an indicator of a resource allocated at tha server for storing the data of each of the files; and

wherein the request sent to the server to read an amount ot data from a lile includes the indicator of the resource allocated for storing the data of the file.

11. The apparatus of claim 8, further comprising:

means for sending the server a list of files which may be accessed by the client;
means for receiving from the server an indicator of a resource allocated at the server for storing the data of each of the files; and

wherein the request sent to the server to read an amount of data from a file includes the indicator of the resource allocated for storing the data of the file.

12. A network interface for controlling flow of data over a computer network from a server to one or more channels at a client, comprising:

a buffer for receiving and storing data packets from the server, wherein each data packet Includes an Indication of one of the channels at the client;
a controller that sends a request to a server indicating one of the channels at the client, a file to be accessed and a desired amount of data to be read from the file and that controls flow of transfer of tha desired amount of data by sending a sequence of messages to a server authorfzing data to be transferred from the file in an amount less than an amount of space available in the buffer on the network Interface; and
means for processing the received data to provide the received data from the buffer to a de-

vice associated with the indicated channel.

13. The network interface of claim 12, wherein the means for processing comprises:

    a first channel having a direct memory access engine for accessing memory in a video output device;
    a second channel having a direct memory access engine for accessing memory in an audio output device;

    wherein the indication of the channel at the client includes an indication of one of the first and second channels; and
    means for transferring the data in each data packet directly from the buffer to the memory of the output device corresponding to the indication of the first and second channels using the direct memory access engine of the channel, thereby bypassing copying of the data by an operating system to a system level memory.

14. A process for controlling flow of data over a computer network from a server to a client, comprising:

    receiving a request from the client indicating a channel at the client, a file to be accessed and a desired amount of data to be read from the file;
    retrieving date from the file to be accessed and storing the data in a buffer;
    receiving a message from the client authorizing an amount of data to be transferred from the file;
    sending data packets to the client, wherein each data packet includes an indication of the channel at the client and data read from the buffer; and
    repeating steps of receiving the message and sending data packets until the amount of data to be read from the file is received.

15. An apparatus for controlling flow of data over a computer network from a server to a client, comprising:

    means for receiving a request from the client indicating a channel at the client, a file to be accessed and a desired amount of data to be read from the file;
    means for retrieving data from the file to be accessed and storing the data in a buffer;
    means for receiving a message from the client authorizing an amount of data to be transferred from the file;
    means for sending data packets to the client, wherein each data packet includes an indication of the channel at the client and data read from the buffer; and

    wherein the means for receiving the message and sending data packets cooperate to transfer to the client the desired amount of data.

16. A process for transferring media data for a plurality of streams from a server over a computer network to a client during synchronized playback of the media data, wherein the client has one or more output devices for playing back each stream, and wherein each stream has a buffer for storing media data before playback, the process comprising:

    determining an amount of data in the buffer for the output device for each stream;
    selecting one of the streams, wherein the selected stream has the least amount of data in the buffer for the output device for the stream, and selecting a desired amount of data to be read for the stream;
    for the selected stream, issuing one or more requests for data from the server over the computer network until the desired amount of data to be read is received from the server;
    processing the data received from the server to provide the received data from the computer network to the buffer for the output device associated with the selected stream for playback by the output device; and
    repeating steps of determining, selecting, issuing and processing during synchronized playback of the media data of the plurality of streams.

17. The process of claim 16, wherein issuing requests comprises:

    sending a request to the server indicating the selected stream at the client, a file to be accessed and the desired amount of data to be read from the file;
    sending a message to a server authorizing data to be transferred from the file in an amount less than an amount of space available in a buffer on the network interface; and
    receiving date packets from the server and storing the data packets into the buffer, wherein each date packet includes an indication of the selected stream at the client.

18. The process of claim 17, wherein one of the output devices is a video output device and another of the output devices is an audio output device, the process further comprising:

    associating the video output device with a first one of the streams, wherein the first stream has

a direct memory access engine;
associating the audio output device with a second one of the streams, wherein the second stream has a direct memory access engine;

wherein the indication of the stream includes an indication of one of the first end second stream; end

wherein processing the data received from the server comprises transferring the data directly from a buffer of the network interface to the buffer essocieted with the output device corresponding to the indication of the first end second streems using the direct memory access engine for the stream, thereby bypassing copying of the data by the operating system to system level memory.

19. The process of claim 18, further comprising:

sending the server e list of files which mey be eccessed by the client;
receiving from the server an indicator of a resource allocated at the server for storing the data of each of the files; and

wherein the request sent to the server to read a desired amount of data from a file includes the indicator of the resource allocated for storing the data of the file.

20. The process of claim 16, wherein one of the output devices is a video output device end another of the output devices is an audio output device, the process further comprising:

associating the video output device with a first one of the streems, wherein the first stream has a direct memory access engine;
associating the audio output device with e second one of the streams, wherein the second stream has a direct memory access engine;

wherein the indicetion of the stream includes an indication of one of the first and second streams; and

wherein processing the date received from the server comprises transferring the data directly from a buffer of the network interface to the buffer associated with the output device corresponding to the indication of the first and second streams using the direct memory access engine of the stream, thereby bypassing copying of the data by the operating system to system level memory.

21. The process of claim 20, further comprising:

sending the server a list of files which may be accessed by the client;
receiving from the server an indicator of a resource allocated at the server for storing the data of each of the files; and

wherein the request sent to the server to read a desired amount of data from a file includes the indicator of the resource allocated for storing the data of the file.

22. The process of claim 16, further comprising:

sending the server e list of files which may be accessed by the client;
receiving from the server an indicator of a resource allocated at the server for storing the date of each of the files; and

wherein the request sent to the server to read a desired amount of data from a file includes the indicator of the resource allocated for storing the data of the file.

23. The process of claim 16, wherein issuing one or more requests comprises issuing requests according to an emount of space available in a buffer of the client.

24. The process of claim 16, wherein issuing one or more requests comprises issuing a request only when space is available in a buffer in the client.

25. The process of claim 16, wherein issuing one or more requests comprises issuing a request so as to ensure that the flow of data is within a capability of the client to receive data for delivery to the buffer for the output device of the selected stream.

26. The process of claim 16, wherein issuing one or more requests comprises issuing a request such that each request permits the server to send only as much data as the client can reliably receive.

27. The process of claim 16, wherein issuing one or more requests comprises issuing a request such that each request permits the server to send only es much data as the client can receive reliably at a full speed of the computer network.

28. The process of claim 16, wherein issuing one or more requests comprises issuing a request such that each request ensures that the server does not transmit data faster than the client can eccept the data.

29. The process of claim 16, wherein issuing one or more requests comprises issuing a request such that each request authorizes the server to send only an amount of data which a buffer in the client is ca-

pable of receiving.

30. A system for transferring media data for a plurality of streams from a server over a computer network to a client during synchronized playback of the media data, wherein the client has one or more output devices for playing back each stream, and wherein each stream has a buffer for storing media data before playback, the process comprising:

    means for determining an amount of data in the buffer for the output device for each stream;
    means for selecting one of the streams, wherein the selected stream has the least amount of data in the buffer for the output device for the stream, and selecting a desired amount of data to be read for the stream;
    means for issuing one or more requests for data from the server over the computer network for the selected stream until the desired amount of data to be read is received from the server;
    means for processing the data received from the server to provide the received data from the computer network to the buffer for the output device associated with the selected stream for playback by the output device; and

    wherein the means for determining, selecting, issuing and processing cooperate during synchronized playback of the media data of the plurality of streams.

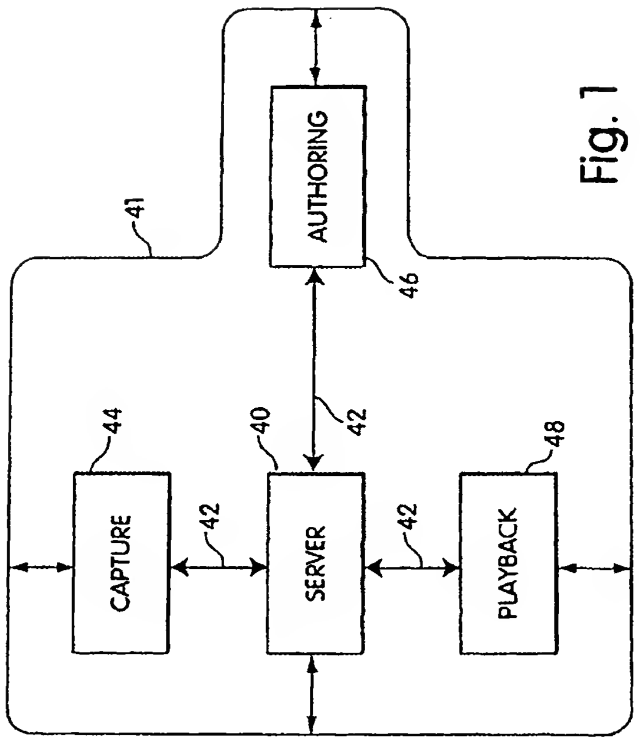


Fig. 1

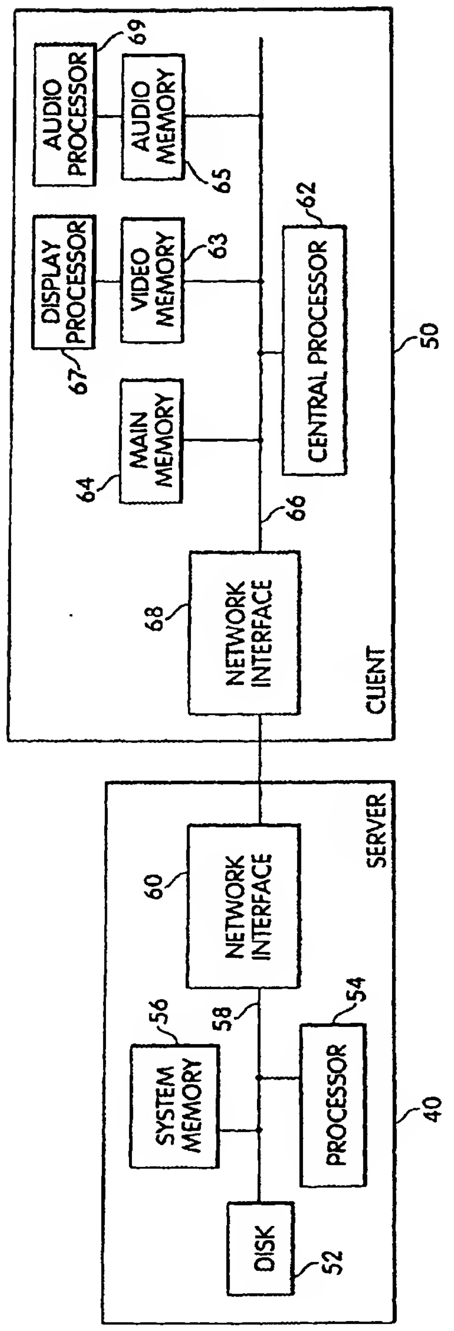

CLIENT
50
AUDIO PROCESSOR
69
AUDIO MEMORY
65
DISPLAY PROCESSOR
67
VIDEO MEMORY
63
CENTRAL PROCESSOR
62
MAIN MEMORY
64
66
NETWORK INTERFACE
68
SERVER
40
NETWORK INTERFACE
60
58
SYSTEM MEMORY
56
PROCESSOR
54
DISK
52

Fig. 2

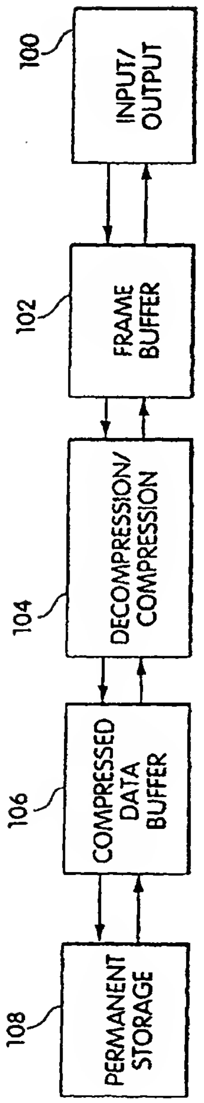


Fig. 3
(Prior Art)

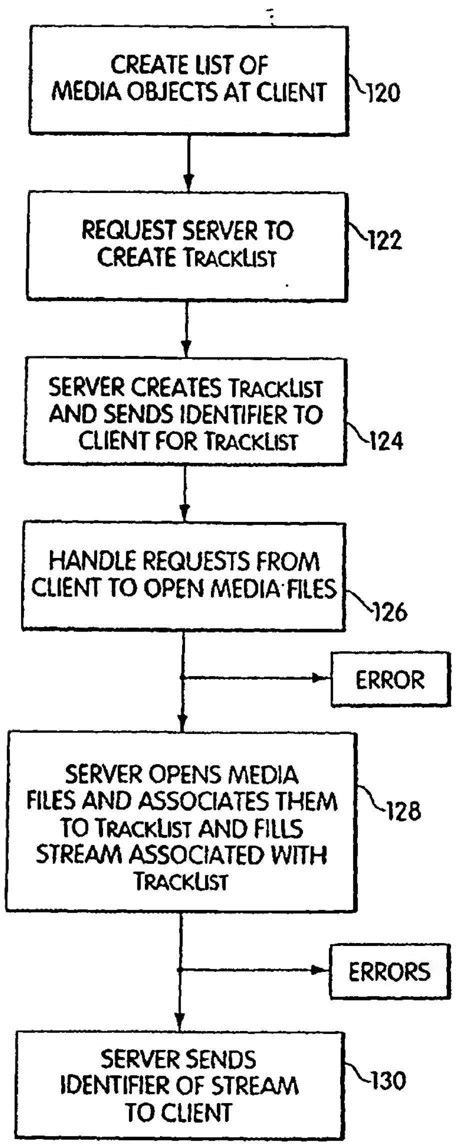

CREATE LIST OF MEDIA OBJECTS AT CLIENT
120
REQUEST SERVER TO CREATE TRACKLIST
122
SERVER CREATES TRACKLIST AND SENDS IDENTIFIER TO CLIENT FOR TRACKLIST
124
HANDLE REQUESTS FROM CLIENT TO OPEN MEDIA FILES
126
ERROR
SERVER OPENS MEDIA FILES AND ASSOCIATES THEM TO TRACKLIST AND FILLS STREAM ASSOCIATED WITH TRACKLIST
128
ERRORS
SERVER SENDS IDENTIFIER OF STREAM TO CLIENT
130

Fig. 4

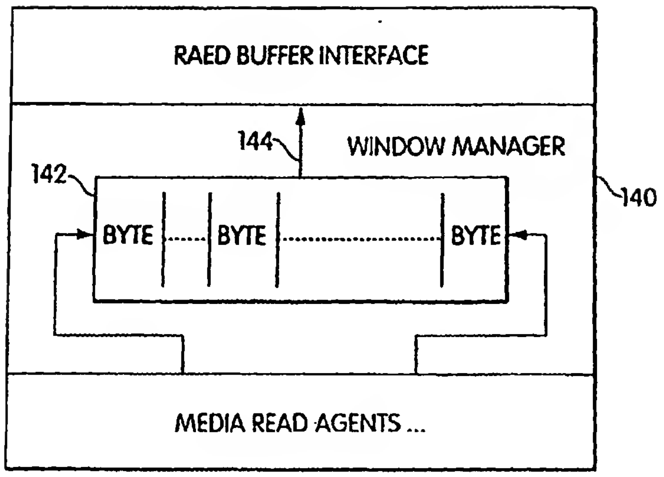

RAED BUFFER INTERFACE
144
WINDOW MANAGER
142
140
BYTE
BYTE
BYTE
MEDIA READ AGENTS ...

Fig. 5

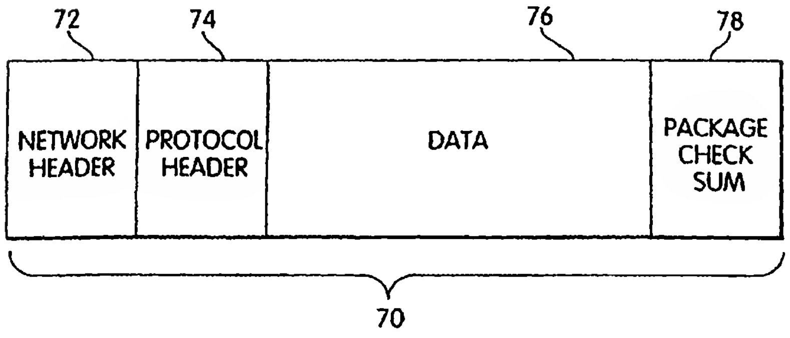
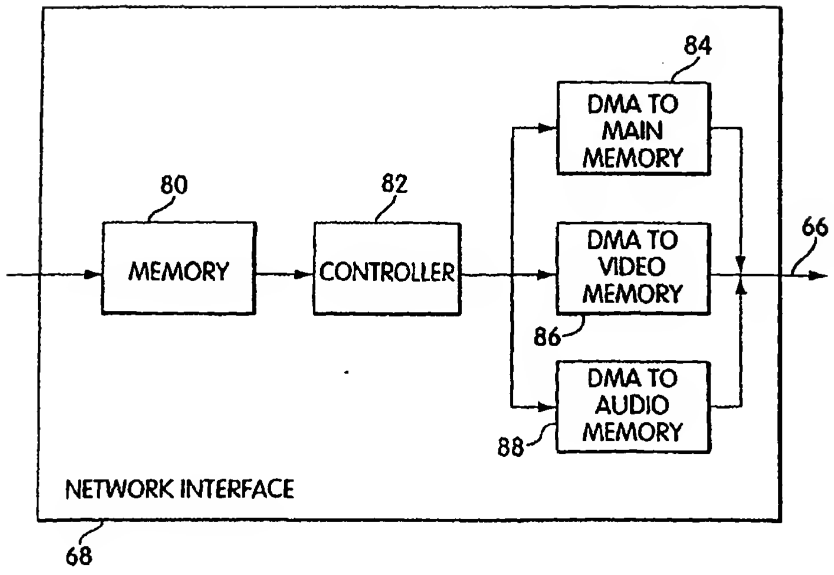
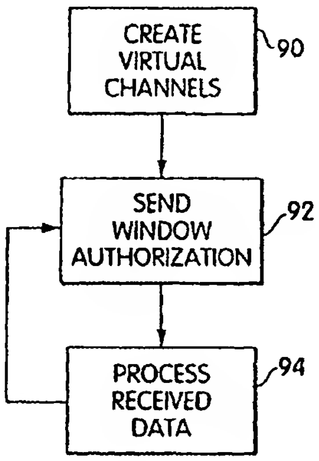

72
74
76
78
NETWORK HEADER
PROTOCOL HEADER
DATA
PACKAGE CHECK SUM
70

Fig. 6



84
DMA TO MAIN MEMORY
80
82
66
MEMORY
CONTROLLER
DMA TO VIDEO MEMORY
86
DMA TO AUDIO MEMORY
88
NETWORK INTERFACE
68

Fig. 7



CREATE VIRTUAL CHANNELS
90
SEND WINDOW AUTHORIZATION
92
PROCESS RECEIVED DATA
94

Fig. 8